Energino: Energy Saving Tips for Your Wireless Network

Roberto Riggio, Karina Mabell Gomez, Tinku Rasheed
CREATE-NET
Trento, Italy
first.lastname@create-net.org

Cigdem Sengul
Telekom Innovation Labs, TU-Berlin
Berlin, Germany
cigdem@net.t-labs.tu-berlin.de

ABSTRACT

The energy wasted in wireless networks is a serious concern and the main challenge lies in determining when and where the energy is wasted. In this demo, we present *Energino*, an energy measurement and control system designed to deliver high performance while remaining a cheap solution.

Categories and Subject Descriptors

C.2.3 [**Computer Communication Networks**]: Network Operations

General Terms

Experimentation, Measurement, Performance

Keywords

Arduino, Energy consumption monitoring, Open hardware, Wireless

1. INTRODUCTION

Current wireless networks are not designed with energy conservation in mind. They generally cater to the worst case scenario, which mandates for an overprovisioning of resources. For instance, the enterprise WLAN in Intel in Portland, OR, has 125 APs over a four floors building with an inter–node distance of 5 m to provide sufficient capacity for four *very close users* using voice, data, and multimedia applications *simultaneously*. However, measurements showed that these APs spend $\approx$ 20 to 80% of their time idling and consume 8.76MWh of energy per year in total [1]. Similar stories repeat in other deployments. The core of the problem is that current networks do not gracefully adjust their energy consumption accordingly to the "amount of work".

In this demo, we show a system, called *Energino* that can be used to both monitor *and* control the energy consumption of WLANs. *Energino* is an Arduino-based plug–load meter designed to monitor, in real-time, the energy consumption of DC devices. Compared to its commercial counterparts, such as Watts Up? [1], *Energino* was shown to provide a cheaper solution with better performance in terms of resolution and sampling rate [2]. This demo presents the prototype of a more complex system with mechanical relays and a controller to adapt the energy consumption of devices under control.

[1]https://www.wattsupmeters.com

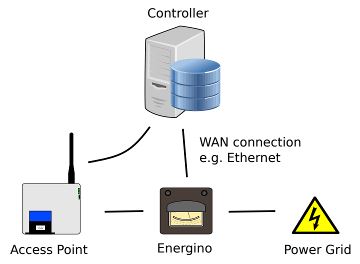


Figure 1: *Energino* network architecture.

We release both the hardware schematics and the software with a permissive BSD license[2] to encourage the research community to use and extend it.

2. THE ENERGINO SYSTEM

In this section, we detail *Energino* main components.

2.1 Energino Devices

To measure and control energy, we extended the Arduino with a custom module integrating a voltage sensor (based on a voltage divider), a current sensor (based on the Hall effect), and a mechanical relay. Each input can be sampled with a period of $100\mu s$, which results in a maximum sampling rate of $10kHz$, using the Arduino 10–bit Analog–to–Digital Converter (ADC). *Energino* can monitor DC loads up to $60V$ and absorbing up to 5A. The resolution for the voltage is 54mV while the resolution for the current is configurable and depends on the maximum current. More specifically, for currents up to 5A the resolution is 26mA, while for currents up to 280mA the resolution is 0.5mA.

During each sampling period the Arduino continuously polls the voltage and the current sensors and accumulates the values into two separate registers. At the end of each period, the average power consumption is computed and the result is sent over the USB interface. This is done in order to filter out fluctuations in the values read from the analog inputs. For example, if the sampling period is set to $1s$, both the voltage and the current readings will be the average of $\approx$ 5000 samples.

2.2 Measurement Framework

The measurement framework conceptually consists of three layers: (1) *The network layer* contains the devices to be monitored. (2) *The energy monitoring layer* contains the *Energino* devices that measure the energy consumption and implement the actions scheduled by the controller. Notice

[2]http://www.wing-project.org/energino/

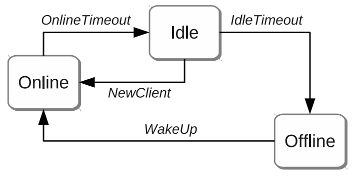


Figure 2: *Energino* finite state machine.

that *Energino* and the APs share both the connection to the Internet and the actual power source. Finally, (3) *the control layer* contains the network controller responsible for gathering network and energy usage statistics from both the network and the energy monitoring layers. The controller also schedules the necessary actions to switch the operating mode of the different network devices. The network architecture is sketched in Fig. 1.

We currently support three operating modes. In the *online* mode, an AP and its wireless interfaces are always on. This is the default mode for a standard WiFi AP. We introduce an *idle* mode, in which the AP is powered on but its wireless interfaces are periodically switched on/off according to a programmable duty-cycle. This mode is similar to the Power-Save Mode typically reserved for client devices in IEEE 802.11. In the *offline* mode, the entire AP is turned off and only the *Energino* is powered.

The controller uses a Finite State Machine to transition the network devices to different modes (see Fig. 2). If the AP has been inactive for at least T_{online} seconds, an *Online-Timeout* event is triggered. Here, inactive means either the AP has no associated clients or, its clients are not sending or receiving. In this case, the AP switches from *Online* to *Idle*. If the AP has been in the *Idle* mode for at least T_{idle} seconds, this generates an *IdleTimeout* event and the AP is switched to *Offline*. In the *Idle* mode, if a new client has just made an association request to the AP or an existing client started traffic, the AP returns to the *Online* mode. Finally, the controller can trigger a *WakeUp* event to power–up an AP, and switch the AP from *Offline* to *Online*.

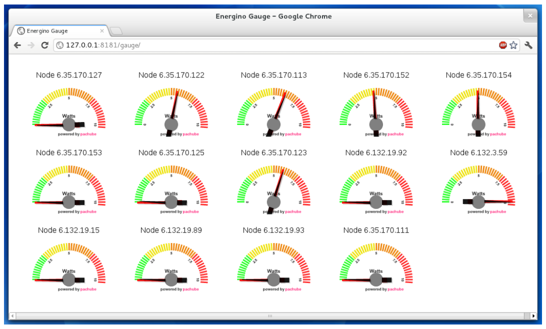
While both the operating modes and the events we support currently are quite general, the framework can be easily extended to using different modes and state machines to realize different energy-saving solutions.

3. DEMO SCENARIO

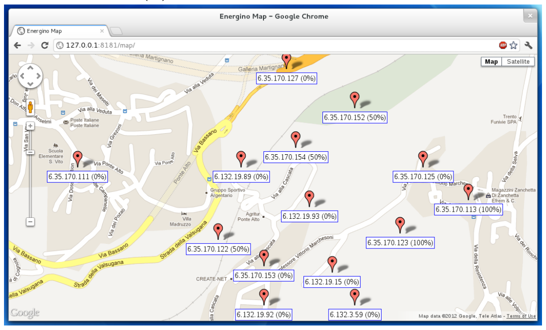
In this demo, we show the operation of *Energino* in a high-density deployment scenario (as the Intel deployment). In the network layer, we will have 3 APs and $3-5$ clients. The energy monitoring layer will be composed of 3 *Energino* devices, and one laptop will act as the controller. We assume the APs form a cluster, and one AP is enough to provide sufficient network coverage at least at the basic transmission rates (i.e., 1/6 Mbps for IEEE 802.11abg and MCS 0 for IEEE 802.11n). This node can be considered as the cluster-head. Within this scenario, we plan to show how the controller monitors and controls the energy consumption of the network. A management backend written in Python is used to both configure *Energino*'s operating parameters, e.g., sampling rate and resolution, and to gather the network and the energy consumption statistics.

In our scenario, the network transitions through different traffic load stages.

- *Low load:* At this stage, all APs, except the cluster-head, are off.



(a) Energy consumption.



(b) Network topology and duty cycle.

Figure 3: The *Energino* monitoring dashboard.

- *Increasing load:* As more clients join the network the controller starts turning on APs. At the beginning, the clients will be able to associate only to the cluster-head, however, as more APs turn on, the clients may re-associate to these APs.
- *Decreasing load:* All APs, except the cluster-head, gradually switch first to *Idle*, then *Offline*, after *Online-Timeout* and *IdleTimeout*, respectively.

Both the energy consumption and the current duty cycle of each APs will be visualized using a web-based dashboard. Fig. 3 show example screenshots of the dashboard, detailing the energy status of the corporate 14-AP WLAN at CREATE-NET. The figures show that the APs with 0% duty cycle (e.g., 6.132.19.89) are powered down, and the APs with the highest duty cycle (e.g. 6.35.170.113) have the highest energy consumption.

4. DISCUSSION

In this demo, we presented an affordable solution for monitoring and controlling the energy consumption of wireless deployments. Our goal is to eventually use *Energino* as a benchmarking tool to evaluate performance of energy saving solutions under different reference traffic conditions seen commonly in wireless networks. This will allow understanding the energy vs. performance trade-offs, and consequently design more performant and low-power solutions.

5. REFERENCES

[1] A. Jardosh et al., "Green WLANs: On-demand WLAN infrastructures," *Mobile Networks and Applications*, vol. 14, no. 6, pp. 798–814, Dec. 2009.

[2] K. M. Gomez et al., "Energino: An hardware and software solution for energy consumption monitoring," in *WinMee*, 2012.